# ENHANCED HEADER COMPRESSION PROFILE

## FIELD OF THE INVENTION

The invention relates to robust header compression. State-of-the-art robust header compression scheme compress the RTP/UDP/IP headers to typically one byte for audio streams and two bytes for video streams. This new scheme compresses the RTP/UDP/IP headers of some video streams to one byte.

## BACKGROUND OF THE INVENTION

The current problem of header compression is that, for some media such as video, both compressed sequence number (SN) and compressed timestamp (TS) have to be sent in most, if not all compressed packets. The invention allows one to send only the SN, and derive TS from SN.

## SUMMARY OF THE INVENTION

The new scheme takes advantage of the pattern observed in the media stream to allow the compressor to get into the highest compression state (SO or second order state) more often.

Other advantages will be readily appreciated, as the invention becomes better understood by reference to the following detailed description and the accompanying drawings.

## BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 shows typical cases of TS as a function of SN for audio;

FIG. 2 shows typical cases of TS as a function of SN for video;

FIG. 3A shows SO pattern, TS as a function of SN;

FIG. 3B shows IP-ID as a function of SN;

FIG. 3C shows M bit as a function of SN; and
FIG. 4 is a logic diagram showing the compressor logical states.

## DETAILED DESCRIPTION OF THE INVENTION

Header compression framework:

This application claims the benefit of U.S. Provisional Application No. 60/236,120, filed September 28, 2000 and U.S. Provisional Application No. 60/239,703 filed October 12, 2000.

For an overview of the robust compression framework, refer to Robust header compression (ROHC), draft-ietf-rohc-rtp-02, draft internet Request For Comments (RFC), which is attached as an appendix hereto and incorporated by reference. This will be referred to hereafter as the current scheme or ROHC. The compression can be in three different states: initialization, first order, second order. In the initialization state, the compressor sends full header packets (no compression). In first order state, it sends FO packets which contain only encoded dynamically changing fields. A typical FO header size is two bytes or more. In SO state, it sends only an encoded SN. The SO state is defined relative to a pattern that a series of header fields follow. There are different modes in which the compression scheme can operate. In reliable mode, the compressor makes sure, through decompressor acknowledgements, that the decompressor is synchronized before going to a higher compression state. In unidirectional or optimistic mode, if the compressor estimates that under all likelihood the decompressor is synchronized, it then goes to a higher compression state. Optimistic and unidirectional mode require that the decompressor be able, for example through the use of a checksum, to check that the compressor and decompressor are actually synchronized.

SO pattern:

The current pattern is defined as:

- second order difference (relative to the SN) of TS being null
- second order difference of IP-ID being null
- All remaining fields being constant.

This proves to be very well suited to audio. For a typical audio stream, the pattern should be verified for the packets generated during a talkspurt (i.e. when actual speech is detected and encoded by the encoder). Since no or very few (comfort noise) packets are usually generated during silence, this

results in the first few packets of a talkspurt to be sent as FO packets and the rest of the packets sent as SO. However, this pattern is not suited to video. This can be seen on figure 1 and 2, which show typical cases of TS as a function of SN for respectively audio and video. In figure 1, it is clear that the second order difference is zero over long period of times. However in figure 2, timestamp jumps are much more frequent (at every frame boundary) and the second order difference is null only over short period of times (the packets belonging to a given frame which all have the same timestamps).

As a consequence, it might happen that in case of video, the compression state is stuck in FO. In the reliable mode, if the link ROUND-TRIP TIME (RTT) (round trip time) is more than the frame skip (ie the time between two coded frames), the compressor can not reliably get in SO (second order) state. Even in the case of a very fast RTT or in optimistic mode, the compression state would go back to FO at every frame boundary.

As can be seen from figure 2, TS is a staircase function of SN. This staircase function could be a pattern used for SO. However to qualify as a pattern, this function has to be fully known. In other words, the length of a step and the jump between steps should be constant over a large enough period of time, i.e. several times the RTT. We argue here that this can be the case, at least when B frames are not used. B frames are considered later in this document.

In effect, payload formats for video encoding are such that a given RTP packet may not contain video data from two different frames. Two consecutive (as generated by the sender application) RTP packets will either have the same RTP timestamp if they carry data from the same picture or their timestamp difference will reflect the time interval between their respective frame sampling instants. Therefore, in order to be predictable, the number of packets per frame and the frame skip should be constant. We show hereafter, that in many cases a video encoder can keep these parameters constant.

For the sake of robustness, video applications can choose to packetize a video frame so that it contains a constant number of macro-bloks or in other words every video frame is sliced identically. This is much as an audio packet

covers a given length of audio. The only good reason (at least we can think of) why the sending application would choose to do otherwise is if it wanted to limit the maximum packet size. This would imply that some frames may have more packets than the usual number of packets per frame. However, the number of such frames will often be limited. For example, an intra frame could be sent in a higher number of packets. This would imply that the compression goes to FO when these packets are compressed.

An encoder is given a target frame rate. As long as the encoder matches its target frame rate, the timestamp jump will be constant. However, encoder implementations usually only match an average target frame rate and the frame skip may be variable. Nevertheless, at least in the streaming case, an encoder may use a higher buffer (higher delay) which provides greater flexibility to the rate control and maintain the target frame-rate throughout the connection (or at least over a long period of time). The invention is expected to be useful for the encoders that have been so designed.

In addition, the RTP marker bit should be set only for the last packet of a frame. This can thus also be derived from the pattern.

We therefore define a pattern as:

□ TS is some function f of SN, where f is more general than just linear extrapolation. For example, f can be a staircase function of SN which has constant length steps and constant jumps between steps.

□ The marker bit is also some function g of the SN. For example, the marker bit is set only for the last packet of each step

□ IP-ID second order difference is null (same as audio)

Using the example of video, this pattern is shown on FIGS. 3a, 3b, 3c for a series of 40 packets where the frame rate is 10 (frames per second) fps and the number of packets per frame is 9.

Compressor and decompressor logic

The only modification to the current scheme introduced by the new pattern is relative to how the compressor and decompressor transitions between the FO and SO state. The compressor is the one making the decision as to which

state to operate. The decompressor follows the compressor decisions. It operates in FO state when receiving FO packets and it operates in SO state when receiving SO packets. The compressor logic is shown on figure 4. The FO state is here conceptually divided up into two sub-states FO_1 and FO_2. The compressor enters FO in FO_1 and moves to FO_2 if a pattern is detected. The compressor goes from FO_2 to SO when the decompressor is synchronized, that is when the decompressor would be able to decompress SO packets.

The issues peculiar to the new scheme are therefore:

□ How the decompressor decompresses packets in SO state

□ How the compressor acquires the pattern

□ How the compressor knows the decompressor is synchronized in order to get into SO state

We examine hereafter each of these issues.

SO packets decompression

In SO state, the decompressor must know the pattern functions f and g, in order to decompress SO packets. Again using the video example, the decompressor must know n, the number of packets per frame and TS_inc the timestamp increment between two frames. In addition, it must have previously successfully decompressed a packet which was a first packet frame.

Let call SN_0, TS_0 the sequence number and timestamp of such a packet. For any incoming SO packet, the decompressor decompresses SN as in the current scheme. If n and m are the quotient and modulo of SN-SN_0 by q, i.e. SN-SN_0=q*n+m, the decompressor computes the packet TS according TS=TS_0+q*TS_inc. The marker bit M is set only if m=n-1. All other fields are obtained as in the current scheme.

Pattern detection

The compressor can determine the function by observation of the stream/learning or API or some other means. Again using the video example, and assuming stream observation, the new pattern requires getting enough packets from the sender before the decision can be made. This in turn requires to buffer a copy of a certain number of past packets.

The pattern could be detected by searching in this buffer for three packets whose (SN,TS,M,IP-ID) are such that the second order IP-ID differences are null and (SN_1,TS_1,M_1) (SN_2,TS_2,M_2) (SN_3,TS_3,M_3) are such that:

SN_2=SN_1+1

TS_3=TS_2

M_1=1

M_2=0

M_3=1

This implies that SN_2 is a first packet frame and TS_inc=TS_2-TS_1 and The number of packets per frame is n=SN_3-SN_2

There is a particular case where the pattern can be detected by two packets (SN_1,TS_1,M_1) (SN_2,TS_2,M_2) such that:

SN_2=SN_1+1

M_1=1

M_2=1

In that case, there is only one packet per frame.

Decompressor synchronization

In the current scheme, the compressor needs only to get two ACKs from the decompressor to make sure that the latter is synchronized. The decompressor derives from the last two received packets the first order differences required to decompress SO packets.

However, with the new pattern, this is not enough. There are several ways the compressor may make sure the decompressor is synchronized. We suggest here three of them.

□ After detecting the pattern, the compressor can explicitly send the pattern functions f and g to the decompressor using in-band signaling. Again using the video example, the compressor sends n and TS_inc, along with an indication that the marker bit is set only for the last packet of each step. The compressor has just then to make sure that the decompressor has received a packet with a marker bit set (first frame packet). It then knows that the decompressor has all the information needed to decompress the packet. After

receiving an FO header carrying the pattern description, the receiver should try to acknowledge packets with the marker bit set so that the compressor can start to send SO packets as soon as possible. Pros: decompressor logic is kept low (no need to perform pattern detection). The compressor does not have to know beforehand if the decompressor is not capable of interpreting the in-band signaling; it can be signaled by a REJECT from the decompressor, with cause "Not recognized". Cons: additional overhead, changes to the current packet format.

□ Alternatively, the compressor can observe the acks received from the decompressor to determine if the decompressor has acquired the pattern functions f and g. The decompressor is also performing pattern detection on the decompressed packets in FO mode. When the pattern is acquired, this will be signaled in the subsequent ACKs sent to the compressor. When the compressor gets enough ACKs to indicate that the pattern has been detected, it can start sending SO packets. Pros: the overhead is kept at a minimum. The in-band signaling format does not have to be standardized. Cons: the decompressor has to perform pattern detection which incurs more complexity and higher delays in the cases where the link loses packets used for detection; The pattern functions have to be standardized. The compressor must also know if the decompressor is capable of detecting the function f (the reception of ACKs alone does not ensure that the decompressor has acquired the function); this could be done by some capability exchange or negotiation.

□ The decompressor performs pattern detection and the compressor also performs pattern detection for the packets which have been acknowledged by the decompressor. In other words, the compressor tries to find if the pattern can be detected using only the packets it is certain the decompressor has received. The decompressor should then choose to acknowledge packets which are known to be enough to detect the pattern, for example the triplet shown above. Pros: no modification needed to the packet format. The same packet formats can be used for the audio pattern and the video pattern. Cons:

extra-complexity, delay before entering SO, reduces the freedom of the decoder to choose whether or not to ACK a packet.

B and PB frames:

We presented the pattern as a typical pattern for video. However, in the case where B frames are used, this pattern is not followed. We don't consider B frames as a typical case for the following reasons:

- B frames are used only by a limited number of codecs. They are not used in MPEG 4 simple profile.
- For low bit-rate video conferencing, B-frames are considered not suitable. This is because the P frame needs to be received before the B-frame can be decoded. For a typical 10 fps frame rate, this would mean an additional 100 ms to the end-to-end delay.

In the case where B-frames are used, there could still be a typical pattern if the encoder chooses to encode a fixed number of B-frames per number of encoded frame, for instance every other frame is a B-frame.

Conclusion:

This scheme could be beneficial to many applications. There is no penalty if an application does not follow the pattern. In addition, if such a scheme was standardized, it could be an incentive for applications to choose a packetization strategy that is header compression friendly, i.e. which follows the SO pattern. In particular, designers of future video application for 3G mobile terminals could take this into account.